Using PROC RANK and PROC UNIVARIATE to Rank or Decile Variables

Jonas V. Bilenas
JP Morgan Chase
PhilaSUG June 23, 2009

CHASE

Outline

- What is PROC RANK?
- Why use PROC RANK?
- Examples.
- What about Weighted Data?
- Disadvantages?

What is PROC RANK?

- PROC RANK computes RANKS for one or more numeric variables across observations in a SAS data set and creates a new SAS data set.
  - You can specify the order of ranks (ascending or descending)
  - There are options to handle ties.
  - You can specify groups to generate percentiles, deciles, quartiles, etc.
- The PROC does not produce any printed output.

Why Use PROC RANK?

- Nonparametric Statistics.
- Decile or group variables:
  - Exploratory analysis, useful in modeling when the variable you wish to predict is binary (i.e.; response, default, attrite) to generate bivariate Ranks and Plots.
  - Certain modeling techniques used in Credit industry build models off of binned (grouped) variables. Examples; WOE, Integer Programing.
- Easier than using MACROS, PROC SORT, and DATA steps.

Simple Example

```
data test;
  input spend @@;
  datalines;
122345 235 12 5677 214 432 121 1567
;
run;

proc rank data=test out=r_test;
  var spend;
run;

proc print data=r_test;
run;
```

Simple Example Output

Obs	spend
1	8
2	4
3	1
4	7
5	3
6	5
7	2
8	6

What Happened to the Raw Spend Variables?

Simple Example: Retain values & Ranks

```
proc rank data=test out=r_test;
   var spend;
   ranks r_spend;
run;
```

Obs	spend	r_spend
1	122345	8
2	235	4
3	12	1
4	5677	7
5	214	3
6	432	5
7	121	2
8	1567	6

How about Descending Order?

```
proc rank data=test out=r_test descending;
  var spend;
  ranks r_spend;
run;
```

Obs	spend	r_spend
1	122345	1
2	235	5
3	12	8
4	5677	2
5	214	6
6	432	4
7	121	7
8	1567	3

Can’t I do this with PROC SORT and DATA STEPS?

- For these simple problems the answer is yes. There is always more than one way to do things in SAS.
- For complicated problems, one should always investigate if there is a PROC in SAS that can handle the problem.

How about TIES in the variable you are ranking?

```
data test;
  input spend @@;
  datalines;
122345 235 12 5677 214 432 121 12 1567 235
;
run;

proc rank data=test out=r_test descending;
  var spend;
  ranks r_spend;
run;

proc print data=r_test;
run;
```

Example #4: How about TIES in the variable you are ranking?

Obs	spend	r_spend
1	122345	1.0
2	235	5.5
3	12	9.5
4	5677	2.0
5	214	7.0
6	432	4.0
7	121	8.0
8	12	9.5
9	1567	3.0
10	235	5.5

PROC RANK uses a DEFAULT TIES=MEAN. The MEAN rank is returned.

I Like TIES=LOW. Assigns the lowest rank in event of ties.

spend	r_spend
122345	1
235	5
12	9
5677	2
214	7
432	4
121	8
12	9
1567	3
235	5

```
proc rank data=test
          out=r_test
          descending
          ties=low;
  var spend;
  ranks r_spend;
run;
```

Other TIES Options

TIES=HIGH

spend	r_spend
122345	1
235	6
12	10
5677	2
214	7
432	4
121	8
12	10
1567	3
235	6

TIES=DENSE

spend	r_spend
122345	1
235	5
12	8
5677	2
214	6
432	4
121	7
12	8
1567	3
235	5

Mathematical Precision in SAS (Example from Mike Zdeb)

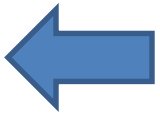
```
data test;
   input fn1 fn2 @@;
   diffa = fn2 - fn1;
   diffb = round(diffa,0.01);
datalines;
0.72 0.68 0.74 0.64 0.63 0.64 0.64 0.69 0.7 0.54 0.56 0.5 0.83
0.78 0.56 0.49 0.65 0.77 0.9 0.87 0.7 0.74 0.85 0.76 0.55 0.56
0.68 0.64 0.64 0.61 0.61 0.66 0.82 0.88 0.66 0.63 0.66 0.71 0.66
0.61 0.59 0.65 0.78 0.63 0.61 0.81 0.82 0.67 0.65 0.76 0.71 0.71
0.74 0.74 0.6 0.85 0.49 0.48 0.54 0.38 0.69 0.67 0.73 0.83 0.67
0.63 0.6 0.54 0.47 0.54 0.42 0.46 0.69 0.64 0.65 0.62 0.57 0.71
0.67 0.79 0.81 0.72
;
run;
```

Mathematical Precision in SAS (Example from Mike Zdeb)

```
proc rank data=test out=ranks ties=low;
   var diffa diffb;
   ranks r_diffa r_diffb;
run;

proc print data=ranks;
run;

proc print data=ranks;
   format diffa diffb 21.18;
run;
```



You can include more than 1 variable in PROC RANK

Mathematical Precision in SAS (Example from Mike Zdeb)

First PROC PRINT:

Obs	fn1	fn2	diffa	diffb	r_diffa	r_diffb
1	0.72	0.68	-0.04	-0.04	16	14
2	0.74	0.64	-0.10	-0.10	5	5
3	0.63	0.64	0.01	0.01	25	25
4	0.64	0.69	0.05	0.05	29	29
5	0.70	0.54	-0.16	-0.16	2	1
6	0.56	0.50	-0.06	-0.06	9	9
7	0.83	0.78	-0.05	-0.05	12	11

Mathematical Precision in SAS (Example from Mike Zdeb)

2nd PROC PRINT:

diffa	diffb	r_diffa	r_diffb
-0.03999999999999900	-0.04000000000000000	16	14
-0.10000000000000000	-0.10000000000000000	5	5
0.01000000000000000	0.01000000000000000	25	25
0.04999999999999900	0.05000000000000000	29	29
-0.15999999999990000	-0.16000000000000000	2	1
-0.06000000000000000	-0.06000000000000000	9	9
-0.04999999999999900	-0.05000000000000000	12	11

Generating Groups with PROC RANK

- **GROUPS option of PROC RANK assigns group values for the non-missing values of the VAR variables.**
- **Group values are stored in the specified RANKS variable.**
- **Values of group values range from 0 to number specified - 1. For example, GROUPS=10 returns 0, 1, 2, .. 9.**
- **TIES: PROC RANK will never split equal variable values across GROUPS. You can end up with unequal group sizes and/or end up with fewer groups than specified by your GROUPS= specification.**
- **Assignment is determined by TIES= option.**
- **DESCENDING option can be used. Higher values of the variables get assigned to lower groups,**

Example of Generating GROUPS

Simulated Data: Higher SCORE predicts higher EVENT (i.e.; response).

```
data test2;
  do i = 1 to 10000;
    score = round(650+rannor(243)*20,1);
    event = (score+rannor(12)*25>700);
    output;
  end;
run;
```

Generating GROUPS:Deciles

```
proc rank data=test2
          out=ranky
          descending
          ties=low
          groups=10;
  var score;
  ranks r_score;
run;
```

Generating GROUPS: Report Results

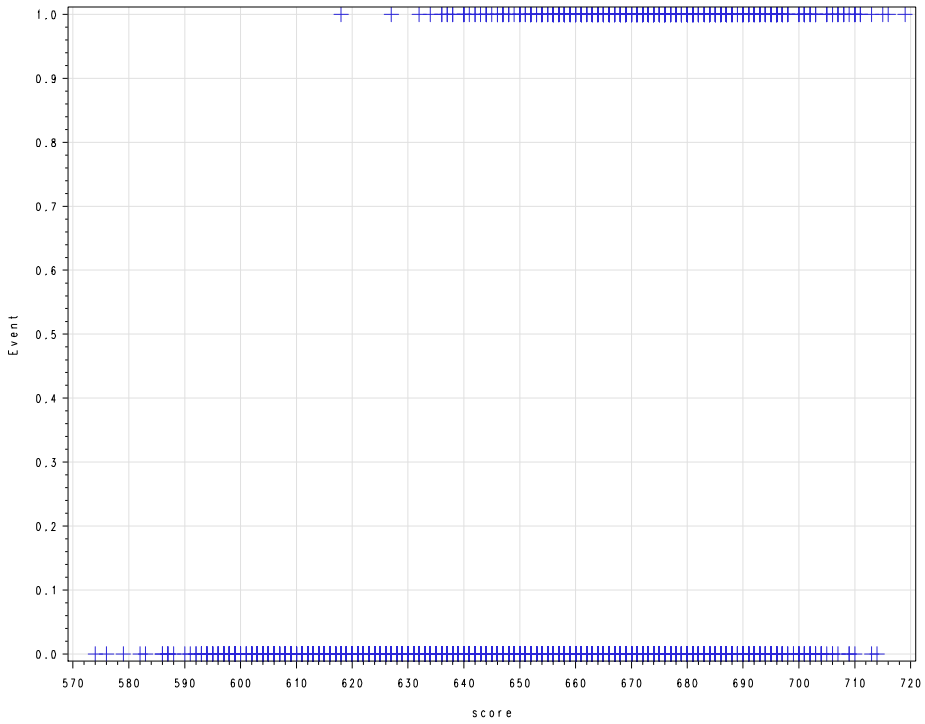
```
proc means data=ranky noprint;
  class r_score;
  var score event;
  output out=report
         n=n  min(score)=score_min
         max(score)=score_max
         mean(event)=event_mean;
run;

proc print data=report noobs;
  var r_score n  score_min score_max event_mean;
  format n comma6.  event_mean percent10.2;
run;
```

Example#1 of Generating GROUPS

r_score	n	score_min	score_max	event_mean
.	10,000	574	719	5.86%
0	1,009	676	719	28.74%
1	1,111	666	675	11.79%
2	982	660	665	6.01%
3	1,094	654	659	4.57%
4	998	649	653	2.20%
5	810	645	648	1.48%
6	1,061	639	644	1.13%
7	975	633	638	0.72%
8	969	624	632	0.21%
9	991	574	623	0.10%

Example of Generating GROUPS: Scatterplot

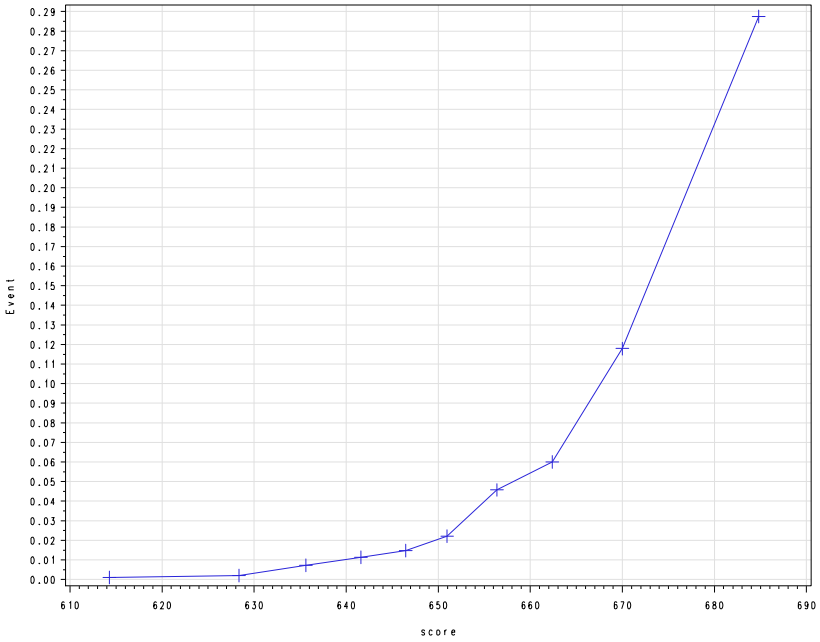



```
filename gout '.';
goptions reset=all
         device=cgmOF97L
         gsfname=gout display
         ;

axis1 label=(angle=90 "Event");
symbol1 i=none v=plus;

proc gplot data=test2;
  plot event*score
       /vaxis=axis1
        grid;
run;quit;
```

GROUPS Scatterplot




```
proc means data=ranky noprint
           nway;
  class r_score;
  var score event;
  output out=plotit mean=;
run;

proc gplot data=plotit;
  plot event*score
       /vaxis=axis1
        grid;
run;quit;
```

GROUPS Scatterplot: LOG ODDS SCALE

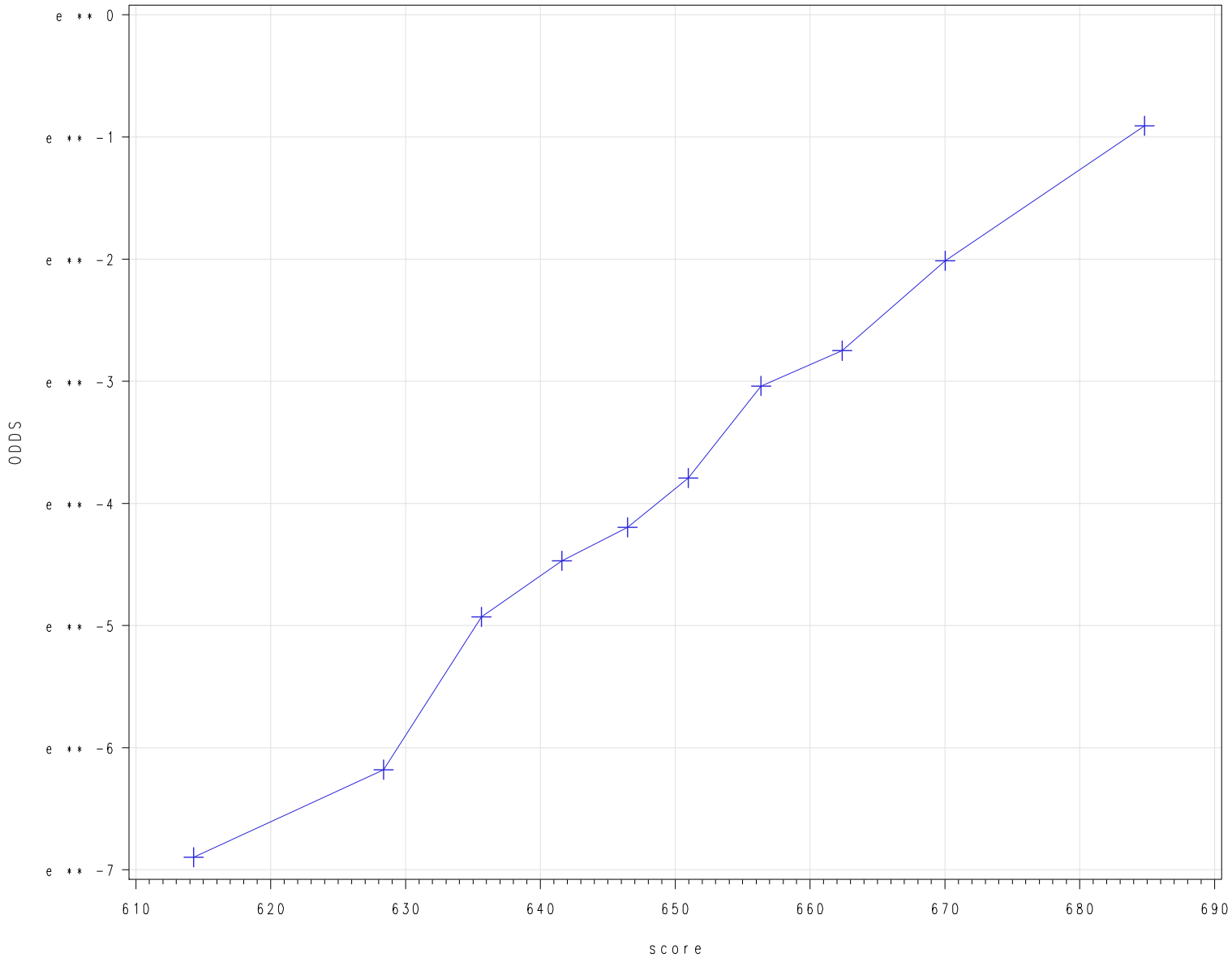
```
data plotit2;
  set plotit;
  odds=event/(1-event);
run;

axis2 label = (angle=90 "ODDS")
      logbase=e logstyle=expand;

symbol1 i=join v=plus;

proc gplot data=plotit2;
  plot odds*score
       /vaxis=axis2
        grid;
run;quit;
```

GROUPS Scatterplot: LOG ODDS SCALE



Sometimes ORDER and TIES Matter

```
data orders;
  do i = 1 to 100;
    x = (i>40);
    output;
  end;
run;

proc rank data=orders out=ranked
          ties=low    groups=2;
  var x;
  ranks r_x;
run;

proc freq data=ranked;
  table r_x;
run;
```

Sometimes ORDER and TIES Matter

Rank for Variable x

r_x	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	100	100.00	100	100.00

Descending

Rank for Variable x

r_x	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	60	60.00	60	60.00
1	40	40.00	100	100.00

Sometimes ORDER and TIES Matter

Default TIES & Ascending

Rank for Variable x

r_x	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	40	40.00	40	40.00
1	60	60.00	100	100.00

P Option in PROC RANK

- P option returns the Percentile in the RANKS Variable (0-100).
- There is also a F option which returns the Fraction (proportion: 0-1) in the RANKS variable.

Generating GROUPS: Getting Cumulative Percents and 2-sample KS Statistic

- 2- Sample Kolmogorov-Smirnov (KS) statistic is often used in evaluating scores that predict a binary response.
  - The KS statistic is a nonparametric statistic to test if there are any distribution differences between 2 samples.

$$KS = \max_j \left| F_1(x_j) - F_2(x_j) \right|$$

- KS statistic can be obtained in PROC NPAR1WAY, but we can also get it from PROC RANK. Let us compare.

2-sample KS Statistic from PROC NPAR1WAY

```
proc npar1way data=test2 D;
  class event;
  var score;
run;
```

Kolmogorov-Smirnov Two-Sample Test (Asymptotic)

D = max |F1 - F2|	0.5463
Pr > D	<.0001
D+ = max (F1 - F2)	0.5463
Pr > D+	<.0001
D- = max (F2 - F1)	0.0000
Pr > D-	1.0000

PROC RANK KS CALCULATION on a Decile

```
proc sort data=test2;
  by event;
run;

proc rank data=test2
          out=events
          p ties=low
          descending;
  by event;
  var score;
  ranks F_score;
run;
```

PROC RANK KS CALCULATION on a Decile

```
proc rank data=events
          out=test3
          groups=10
          ties=low
          descending;
  var score;
  ranks D_score;
run;
```

TEST3 Variables:

- score
- event
- F_score: Cumulative % by EVENT groups.
- D_score: Deciles of score.

PROC RANK KS CALCULATION on a Decile

```
proc means data=test3 noprint nway;
  class D_score;
  var F_score;
  output out=event max(F_score)=F_event
         n=n_event;
  where event;
run;

proc means data=test3 noprint nway;
  class D_score;
  var F_score;
  output out=nonevent max(F_score)=F_nonevent
         n=n_nonevent;
  where event=0;
run;
```

PROC RANK KS CALCULATION on a Decile

```
proc format;
  picture pct (round) low-high = '009.99%';
run;
data ks;
  merge event nonevent;
  by D_score;
  KS=abs(F_event - F_nonevent);
  n=n_event + n_nonevent;
run;
proc print data=ks noobs label;
  var D_score n F_event F_nonevent ks;
  format n comma6. F_event F_nonevent KS pct.;
  label D_score='Decile'
        F_event='Cumulative Event'
        F_nonevent='Cumulative Non Event';
run;
```

PROC RANK KS CALCULATION on a Decile

Decile	n	Cumulative Event	Cumulative Non Event	KS
0	1,009	46.08%	6.75%	39.33%
1	1,111	69.62%	16.84%	52.79%
2	982	80.20%	25.98%	54.22%
3	1,094	89.25%	36.73%	52.52%
4	998	94.20%	47.20%	47.00%
5	810	96.25%	55.53%	40.71%
6	1,061	97.78%	67.31%	30.47%
7	975	99.49%	77.70%	21.78%
8	969	99.83%	88.64%	11.18%
9	991	100.00%	100.00%	0.00%

FROM NPAR1WAY: D = max |F1 - F2| 0.5463

PROC RANK KS: Graph Results by Score

```
filename gout '.';
goptions reset=all
         device=cgmOF97L gsfname=gout display
         ;
proc sort data=events;
  by event descending score  F_score;
run;

data events;
  set events;
  score=score*-1;
run;
```

I want to graph X-axis from High score to Low score

PROC RANK KS: Graph Results by Score

```
%let splits = %str(output out=p pctlpre=P_ pctlpts=10
   to 100  by 10);

proc univariate data=events noprint;
   var score;
   &splits;
 run;

proc transpose data=p out=pt;
run;
```

NAME	COL1
P_10	-676
P_20	-666
P_30	-660
P_40	-654
P_50	-649
P_60	-645
P_70	-639
P_80	-633
P_90	-624
P_100	-574

PROC RANK KS: Graph Results by Score

```
proc sql;
   select distinct COL1
      into :href separated by ' '
      from pt;
run; quit;

proc format;
  picture score (round notsorted) low - 0 = '999‘
  ;
  value event 0='Non-Event' 1='Event'
  ;
run;
```

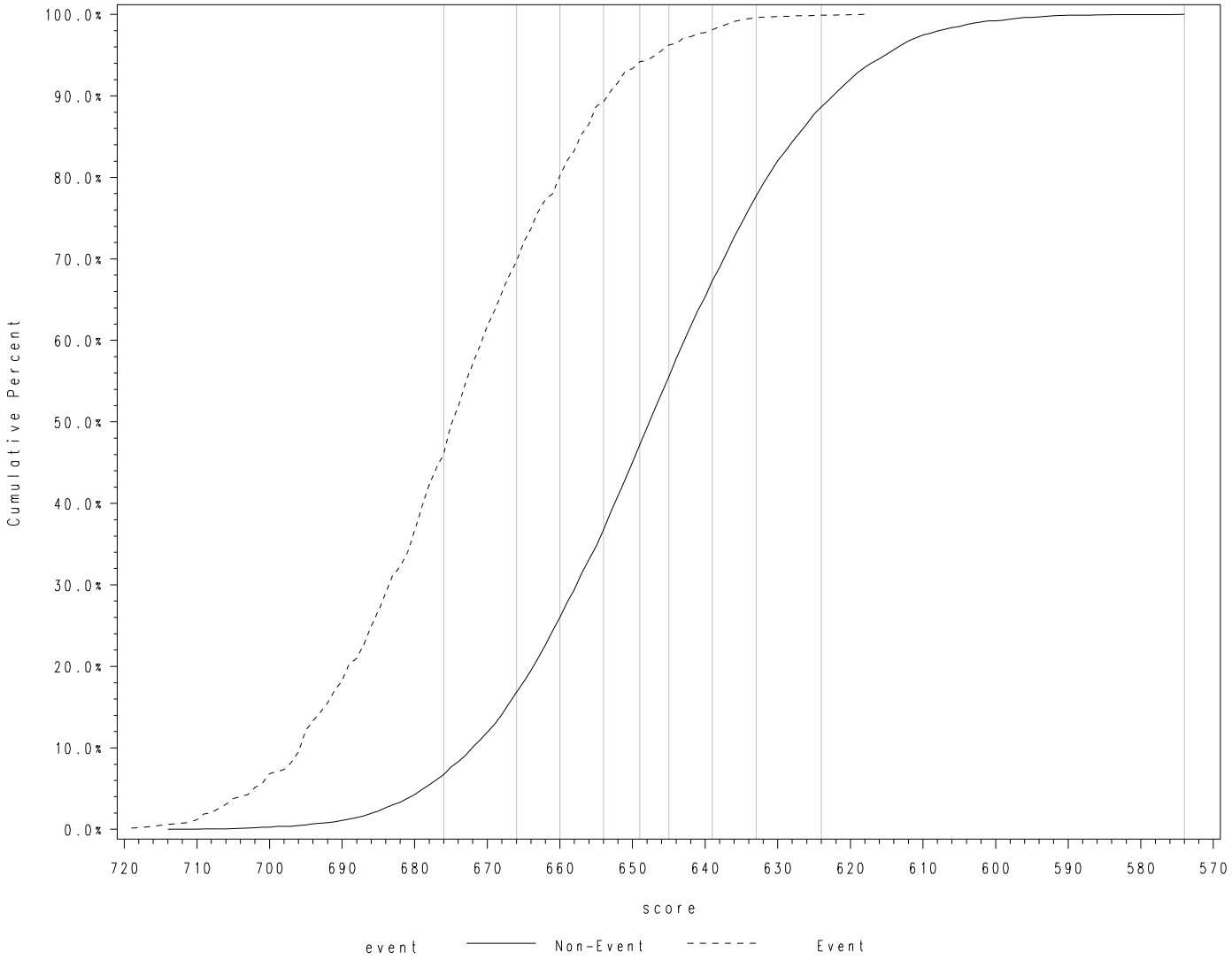
PROC RANK KS: Graph Results by Score

```
symbol1 i=join v=none c=black l=1;
symbol2 i=join v=none c=black l=2;

axis1 label = (angle=90 "Cumulative Percent");

proc gplot data=events;
  plot F_score*score=event
       /vaxis=axis1
        href=&href
        chref=ltgray;
  format score score. F_score pct. event event.;
run;
```

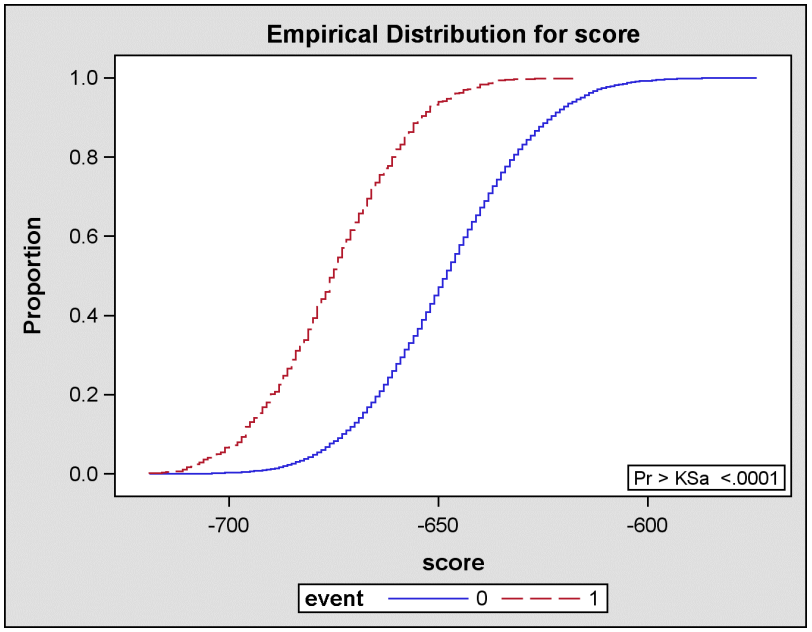
PROC RANK KS: Graph Results by Score



Get PLOT from NPAR1WAY & ODS GRAPHICS

```
ods html image_dpi=300;
ods graphics on / width=4.5in height=3.5in;

proc npar1way D plots=edfplot
                data=events;
  class event;
  var score;
run;
ods graphics off;
```



Weighted RANKS

- PROC RANK does not have a WEIGHT or FREQ Statements.
- PROC NPAR1WAY has a FREQ statement but sometimes our samples are funky and our weights are not integers.
- PROC UNIVARIATE is the solution.

Weighted RANKS: Generate the Sampled Data

```
data test2;
  do i = 1 to 10000;
    score = round(650+rannor(243)*20,1);
    neg_score = score*-1;
    event = (score+rannor(12)*25>700);
    noa=1;
    r_score = neg_score;
    output;
  end;
run;

proc surveyselect data=test2 (where=(event=1)) out=events
   seed=62309 method=srs rate=1 stats;
run;

proc surveyselect data=test2 (where=(event=0)) out=event0
   seed=62309 method=srs rate=.02 stats;
run;

proc append base=events data=event0 force;
run;
```

Weighted RANKS: UNIVARIATE

```
The SURVEYSELECT Procedure

Selection Method    Simple Random Sampling

Input Data Set                TEST2
Random Number Seed            62309
Sampling Rate                     1
Sample Size                     586
Selection Probability             1
Sampling Weight                   1
Output Data Set              EVENTS

The SURVEYSELECT Procedure

Selection Method    Simple Random Sampling

Input Data Set                TEST2
Random Number Seed            62309
Sampling Rate                  0.02
Sample Size                     189
Selection Probability      0.020076
Sampling Weight            49.80952
Output Data Set              EVENT0
```

Weighted RANKS: UNIVARIATE

```
%let splits = %str(output out=p pctlpre=P_
                   pctlpts=10 to 100  by 10);

proc univariate data=events noprint;
  var neg_score;
  &splits;
  weight SamplingWeight;
run;

proc transpose data=p out=pt;
run;

proc sort data=pt
     nodupkey force noequals;
  by  COL1;
run;
```

NAME	COL1
P_10	-676
P_20	-665
P_30	-660
P_40	-655
P_50	-650
P_60	-645
P_70	-640
P_80	-632
P_90	-623
P_100	-588

Weighted RANKS: Generate Format

```
data cntlin;
  set pt end=eof;
  length HLO SEXCL EEXCL $1 LABEL $2;
  retain fmtname "score" type 'N' end;
  nrec+1;
  if nrec=1 then do;
    HLO='L'; SEXCL='N'; EEXCL='Y'; start=.; end=COL1; label=put(nrec-1,z2.);
   output;
  end;
  else if not eof then do;
    HLO=' '; SEXCL='N'; EEXCL='Y'; start=end; end=COL1; label=put(nrec-1,z2.);
   output;
  end;
  else if eof then do;
    HLO='H'; SEXCL='N'; EEXCL='N'; start=end; end=.; label=put(nrec-1,z2.);
   output;
  end;
run;

proc format cntlin=cntlin fmtlib;
run;
```

Weighted RANKS: FORMAT Output

FORMAT NAME: SCORE LENGTH: 2 NUMBER OF VALUES: 10
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 2 FUZZ: STD

START	END	LABEL (VER. V7|V8 04JUL2009:18:13:49)
LOW	-676<00	
-676	-665<01	
-665	-660<02	
-660	-655<03	
-655	-650<04	
-650	-645<05	
-645	-640<06	
-640	-632<07	
-632	-623<08	
-623	HIGH	09

Weighted RANKS: Report

```
proc tabulate data=events noseps formchar='                    ';
  class r_score;
  format r_score score.;
  var score event noa;
  weight SamplingWeight;
  keylabel sum=' ' pctsum=' ‘ mean=‘ ‘;
  table r_score='Ranked Score' all
        ,
        noa='Weighted N'*f=comma8.
        noa='%'*pctsum<r_score all>*f=pct.
        score*(min max)*f=3.
        event='event rate'*mean*f=percent8.2
        /rts=10 row=float misstext=' ';
  title UNIVARIATE;
run;
```

Weighted RANKS: Report Output

UNIVARIATE

	Weighted N	%	score Min	score Max	event rate
Ranked Score					
00	966	9.66%	677	719	27.84%
01	899	8.99%	666	676	16.90%
02	945	9.45%	661	665	5.08%
03	897	8.97%	656	660	5.58%
04	1,272	12.72%	651	655	2.12%
05	764	7.64%	646	650	2.22%
06	1,204	12.04%	641	645	0.75%
07	957	9.57%	633	640	1.15%
08	998	9.98%	624	632	0.20%
09	1,097	10.97%	588	623	0.09%
All	10,000	100.00%	588	719	5.86%

Weighted RANKS: Does it work with ties?

```
if 630<= score <= 650 then score=round(630,1);
```

	Weighted N	%	score Min	score Max	event rate
Ranked Score					
00	989	9.89%	676	719	29.31%
01	781	7.81%	670	675	10.50%
02	1,129	11.29%	663	669	7.09%
03	1,066	10.66%	656	662	6.29%
04	626	6.26%	651	655	4.31%
05	4,358	43.58%	623	630	0.30%
06	1,050	10.50%	593	622	0.10%
All	10,000	100.00%	593	719	5.60%

Weighted RANKS: Does it work with ties?

```
if score > 650 then score=650;
```

score

	Weighted N	%	Min	Max	event rate
Ranked Score					
01	5,743	57.43%	646	650	9.80%
02	1,204	12.04%	641	645	0.75%
03	957	9.57%	633	640	1.15%
04	998	9.98%	624	632	0.20%
05	1,097	10.97%	588	623	0.09%
All	10,000	100.00%	588	650	5.86%

Weighted RANKS: Alternative to FORMAT from Wensui Liu

```
%macro rank(data = , var = , weight = , groups = ,
   output =, round= );

proc univariate data = &data noprint;
  var &var;
  weight &weight;
  output out = _tmp2
  pctlpre = decile_
  pctlpts = 0 to 100 by %sysevalf(100 / &groups);
run;

data _tmp2;
  set _tmp2;
  decile_100 = 9e9;
run;
```

Weighted RANKS: Alternative to FORMAT from Wensui Liu

```
data &output;
  set  &data;
  if _n_ = 1 then do;
    set _tmp2;
  end;

  array d[*] decile_:;
  rank = 999;
  do i = 2 to dim(d);
    if &var < d[i] and rank = 999 then do;
      lower  = round(d[i - 1], &round);
      upper  = round(d[i], 1) - &round;
      rank = i - 1;
    end;
  end;
run;
```

Weighted RANKS: Alternative to FORMAT from Wensui Liu

```
proc tabulate data=&output noseps formchar='            ';
  class rank;
  var score event noa;
  weight &Weight;
  keylabel sum=' ';
  table rank='Rank'
        all
        ,
        noa='Weighted N'*f=comma8.
        noa='%'*pctsum<rank all>=' '*f=pct.
        score*(min max)*f=3.
        event='event rate'*pctsum<noa>=' '*f=pct.
        /rts=10 row=float misstext=' ';
  title UNIVARIATE;
run;
%mend rank;
%rank(data=events, var = score, weight = SamplingWeight, groups =
   10, output = two, round=1);
```

Weighted RANKS: Alternative to FORMAT from Wensui Liu

	score				
	Weighted N	%	Min	Max	event rate
Rank					
1	997	9.97%	588	621	0.10%
2	898	8.98%	623	631	0.11%
3	954	9.54%	632	639	0.84%
4	1,058	10.58%	640	644	1.13%
5	959	9.59%	645	649	1.36%
6	1,075	10.75%	650	654	2.70%
7	939	9.39%	655	659	4.47%
8	1,095	10.95%	660	664	4.47%
9	988	9.88%	665	675	14.27%
10	1,037	10.37%	676	719	27.96%
All	10,000	100.00%	588	719	5.86%

OTHER PROC RANK OPTIONS

- **NORMAL=BLOM | TUKEY | VW**
  - Computes Normal Scores from ranks.
- **SAVAGE**
  - Computes Exponential Scores from ranks.
- **NPLUS1**
  - computes fractional ranks by dividing each rank by the denominator n+1, where n is the number of observations that have nonmissing values of the ranking variable for TIES=LOW, TIES=MEAN, and TIES=HIGH. For TIES=DENSE, n is the number of observations that have unique nonmissing values.

Disadvantage with PROC RANK?

- **Variables and Ranks must fit in memory.**
  - I rarely run into an issue.
  - Don’t rank **all** numeric variables in your data set in 1 PROC RANK.
  - On rare occasions you may need to modify –memsize and –realmemsize SAS batch options.

References

- **How SAS stores Numeric Data**
  - NESUG 2008
    Is .1+.2 equal to .3?
    Shaoji Xu, Pro Unlimited
- **PROC FORMAT**
  - Bilenas, Jonas (2005). The Power of PROC FORMAT, SAS Press.
- **PROC NPAR1WAY**
  - Garavaglia, Susan B. (1999). Applying Familiar Non-Parametric Tests to Evaluation of Neural Network Inputs, NESUG 1999

Thanks

- **Yan Jiang & Haiyan Weng**
  - First to bug me about how to handle WEIGHTS in Ranking.
- **WenSui Liu & Mike Zdeb**
  - Always providing me with new an interesting ways of doing things in SAS.

Disclaimers

- **Your comments and questions are valued and encouraged. Contact the author at:**

Jonas V. Bilenas

JP Morgan Chase Bank

Wilmington, DE 19801

Email: Jonas.Bilenas@chase.com

jonas@jonasbilenas.com

- **SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.**
- **This work is an independent effort and does not necessarily represent the practices followed at JP Morgan Chase Bank.**